National College of Ireland

Comparative performance of RF and GBM for short-term customer segmentation forecasting

MSc Research Project
Cloud Computing

Thomas Jose
Student ID: x22146962

School of Computing
National College of Ireland

Supervisor: Dr Giovani Estrada

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Thomas Jose
Student ID:	x22146962
Programme:	Cloud Computing
Year:	2024
Module:	MSc Research Project
Supervisor:	Dr Giovani Estrada
Submission Due Date:	27/05/2024
Project Title:	Comparative performance of RF and GBM for short-term customer segmentation forecasting
Word Count:	XXX
Page Count:	8

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Thomas Jose
Date:	26th May 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Comparative performance of RF and GBM for short-term customer segmentation forecasting

Thomas Jose
x22146962

1 Introduction

This manual is presented as supplementary material to the thesis report to help other researchers with the setting up of the software environment and other necessary tools to carry out the execution of this research project on their own, without any expert help or intervention.

2 Local PC Configuration

- Intel/AMD based Quad Core PC Configuration
- min 16GB RAM
- min 256GB SSD storage
- Windows 10,11 or Ubuntu 20.04 or higher

3 Software Packages

- Python 3.8 or higher
- Pandas
- Matplotlib
- NumPy
- Seaborn
- SciPy
- Scikit-learn

4 Google Colab - Cloud Computing Platform

Google Colab [1] is a great place to work with deep learning tools like PyTorch, Keras, TensorFlow, and OpenCV. In Colab, you can create notebooks, save notebooks for future runs, share notebooks, and mount your Google Drive so you can use everything that's in it.

4.1 Setting up Google Drive

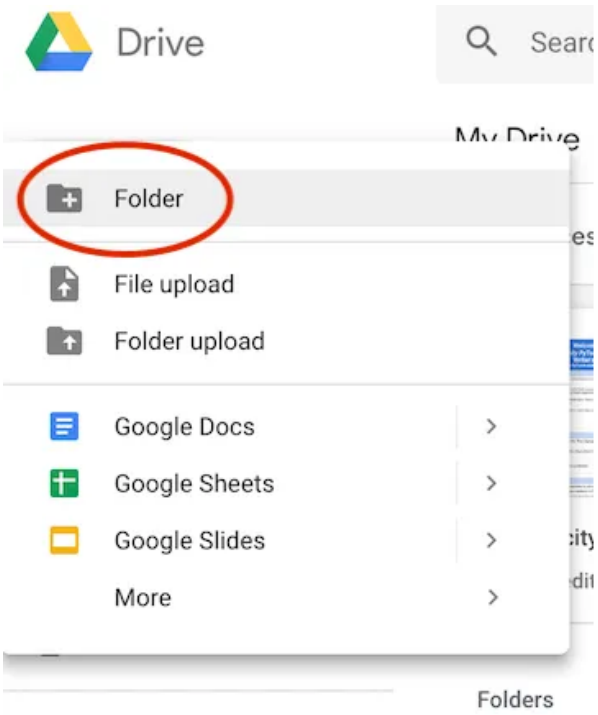

Figure 1: Create Folder in Google Drive

[1] https://colab.research.google.com/

4.2 Creating Colab Notebook

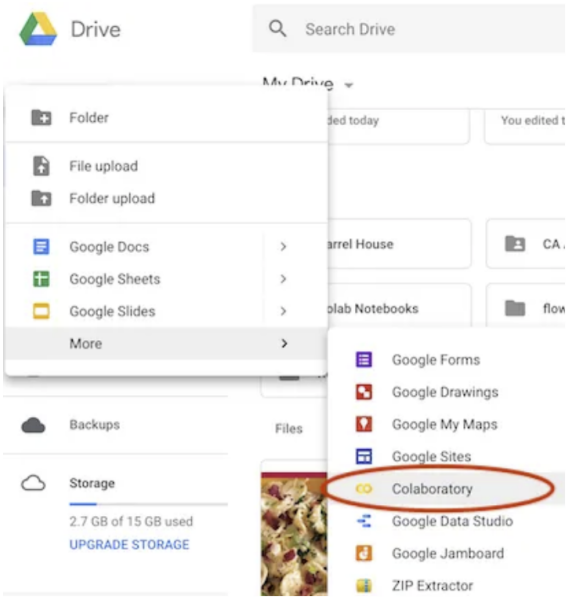

Figure 2: Creating a new jupyter notebook on Colab

4.3 Notebook Compute Instance Selection

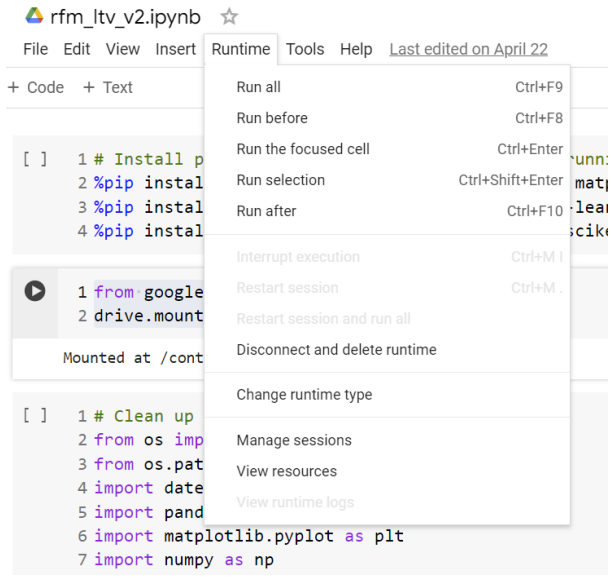

Figure 3: Selecting Runtime Instance

4.4 Runtime Types

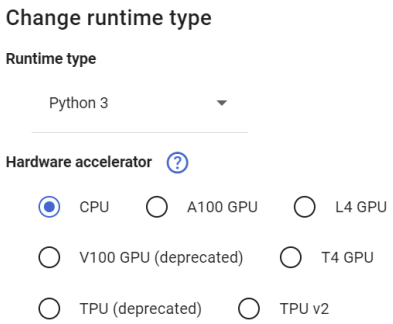

Figure 4: Runtime Types

4.5 Mounting Google Drive for file access

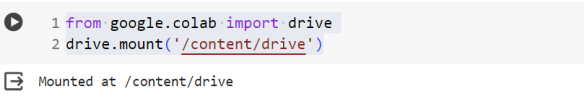

Figure 5: Mount Google Drive

4.6 Runtime Resource Utilization

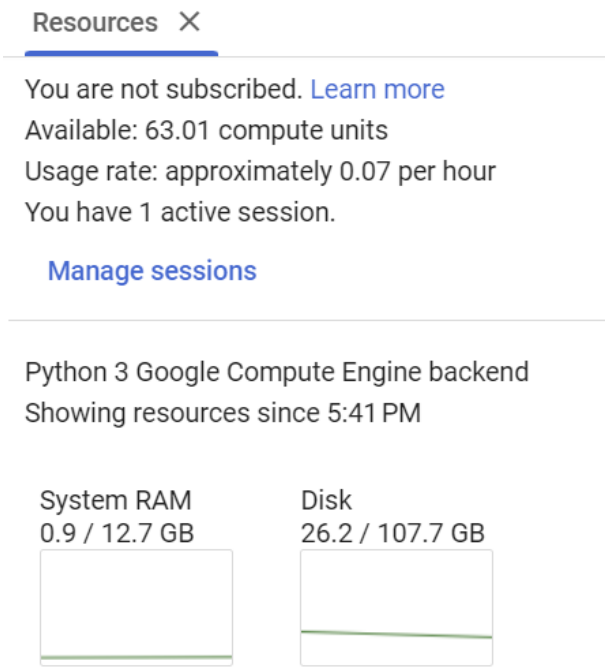

Figure 6: Resource Monitoring

5 Code Development

The code development is carried out on Google Colaboratory using Jupyter Notebooks with preconfigured python and machine learning framework for faster code initialization and execution with better runtime computes.

5.1 Import Libraries

This section specifies the various libraries and packages used in code development that includes Pandas, a data manipulation library, MatplotLib, for creating plots and figures, and NumPy, a numerical computing package for Python. The 'Sklearn' or 'Scikit-Learn' is the machine learning library that makes both Random Forest (RF) and Gradient Boost Machines (GBM) classification possible. The 'datetime' library provides date and time classes to utilize in this work.

```
# Clean up unused imports ...
from os import getcwd
from os.path import isfile
import datetime as dt
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from matplotlib import pylab
import sklearn

from sklearn.ensemble import GradientBoostingClassifier, RandomForestClassifier #, etc, etc
from sklearn.metrics import classification_report
from dateutil.relativedelta import relativedelta
```

Figure 7: Import necessary libraries

5.2 Loading Dataset

This code section loads the online retail dataset from the provided URL as a pandas dataframe, 'retail_data'. The retail_data dataframe is displayed in Figure 9.

```
#############################################################################
# Loading dataset
#
if isfile('Online%20Retail.xlsx'):
    print("Loading dataset")
    retail_data = pd.read_excel('Online%20Retail.xlsx')
else:
    print("Saving dataset to " + getcwd())
    url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/00352/Online%20Retail.xlsx'
    retail_data = pd.read_excel(url)

retail_data.head
```

Figure 8: Load Dataset from source

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom

Figure 9: Online Retail Dataset

5.3 Dataset Preprocessing

This code section performs the data preprocessing analysis on the retail_data dataframe. The customerID columns with NULL values are dropped, and converts the column to integer datatype. The 'Quantity' column is also converted to integer datatype. 'InvoiceDate' and 'InvoiceDate2' are converted to datetime datatype using pd.to_datetime(). The 'TotalPrice' column is created by multiplying 'Quantity' and 'UnitPrice' columns element-wise.

	Quantity	InvoiceDate	UnitPrice	CustomerID	InvoiceDate2	TotalPrice
count	406789.000000	406789	406789.000000	406789.000000	406789	406789.000000
mean	12.028359	2011-07-10 16:28:44.845459968	3.460811	15287.795830	2011-07-10 16:28:44.845459968	20.403860
min	-80995.000000	2010-12-01 08:26:00	0.001000	12346.000000	2010-12-01 08:26:00	-168469.600000
25%	2.000000	2011-04-06 15:02:00	1.250000	13954.000000	2011-04-06 15:02:00	4.200000
50%	5.000000	2011-07-31 11:46:00	1.950000	15152.000000	2011-07-31 11:46:00	11.100000
75%	12.000000	2011-10-20 13:06:00	3.750000	16791.000000	2011-10-20 13:06:00	19.500000
max	80995.000000	2011-12-09 12:50:00	38970.000000	18287.000000	2011-12-09 12:50:00	168469.600000
std	247.927842	NaN	69.318561	1713.573064	NaN	427.612692

Figure 10: Data Preprocessing

	Quantity	InvoiceDate	UnitPrice	CustomerID	InvoiceDate2	TotalPrice
count	406789.000000	406789	406789.000000	406789.000000	406789	406789.000000
mean	12.028359	2011-07-10 16:28:44.845459968	3.460811	15287.795830	2011-07-10 16:28:44.845459968	20.403860
min	-80995.000000	2010-12-01 08:26:00	0.001000	12346.000000	2010-12-01 08:26:00	-168469.600000
25%	2.000000	2011-04-06 15:02:00	1.250000	13954.000000	2011-04-06 15:02:00	4.200000
50%	5.000000	2011-07-31 11:46:00	1.950000	15152.000000	2011-07-31 11:46:00	11.100000
75%	12.000000	2011-10-20 13:06:00	3.750000	16791.000000	2011-10-20 13:06:00	19.500000
max	80995.000000	2011-12-09 12:50:00	38970.000000	18287.000000	2011-12-09 12:50:00	168469.600000
std	247.927842	NaN	69.318561	1713.573064	NaN	427.612692

Figure 11: Preprocessed Dataset

5.4 RFM Score Computation

This code segment performs the RFM analysis by assigning scores to each customer based on their recency, frequency and monetary value. The records are grouped by 'CustomerID', following which 'InvoiceDate' is used to calculate the cutoff date from 'allRecords' (max or most recent invoice date) to derive the 'Recency' value, Counting the number of invoices per customer gives the 'frequency' number, sum of 'TotalPrice' of all purchases for each customer gives the 'MonetaryValue'. A new dataframe, 'rfm' is constructed from this. The percentile ranks are calculated for 'Recency', 'Frequency' and 'MonetaryValue' and the rfm score is assigned to each customer using their percentile ranking using 'pd.qcut()' that creates equal sized bins. A weighted RFM score is calculated using the formula: "0.15 * recency score + 0.28 * frequency score + 0.57 * monetary score".

```
allRecords = retail_data['InvoiceDate'].max()   # a cut off date, dt.datetime(y,m,d). Here we take all records.
rfm = retail_data.groupby('CustomerID').agg({'InvoiceDate': lambda x: (allRecords - x.max()).days,
                                             'InvoiceNo': 'count',
                                             'TotalPrice': 'sum',
                                             'InvoiceDate2': 'max'}).reset_index()
rfm.columns = ['CustomerID', 'Recency', 'Frequency', 'MonetaryValue', 'lastPurchase']

rfm = rfm[rfm['MonetaryValue']>0]  # Remove any customers that have zero contributions ...

# Number of days since a customer last made a purchase (How recently)
rfm['r_percentile'] = rfm['Recency'].rank(pct=True,ascending=True) # the more recent the better
rfm['r_score'] = pd.qcut(rfm['r_percentile'], levels, labels=range(1, levels+1))

#  How often a customer makes a purchase
rfm['f_percentile'] = rfm['Frequency'].rank(pct=True,ascending=False) # the more purchases the better
rfm['f_score'] = pd.qcut(rfm['f_percentile'], levels, labels=range(1, levels+1))

# How much money a customer spends on purchases
rfm['m_percentile'] = rfm['MonetaryValue'].rank(pct=True,ascending=False) # the more spent the better
rfm['m_score'] = pd.qcut(rfm['m_percentile'], levels, labels=range(1, levels+1))

#rfm_scores = rfm[['CustomerID', 'r_score', 'f_score', 'm_score']]
rfm['score'] = rfm['r_score'].astype(str) + rfm['f_score'].astype(str) + rfm['m_score'].astype(str)
print('We have', len(rfm.score.unique()), 'different RFM rankings (from a max of', levels*levels*levels,')')

# Weighted RFM score (typical weights are 0.15r + 0.28f + 0.57m). Smaller score means better customers
rfm.r_score = pd.to_numeric(rfm.r_score)
rfm.f_score = pd.to_numeric(rfm.f_score)
rfm.m_score = pd.to_numeric(rfm.m_score)
rfm['rfm_score'] = 0.15*rfm.r_score + 0.28*rfm.f_score + 0.57*rfm.m_score
```

Figure 12: RFM Analysis

We have 121 different RFM rankings (from a max of 125)

	CustomerID	Recency	Frequency	MonetaryValue	lastPurchase	r_percentile	r_score	f_percentile	f_score	m_percentile	m_score	score	rfm_score
1	12347	1	182	4310.00	2011-12-07 15:52:00	0.035053	1	0.121703	1	0.073808	1	111	1.00
2	12348	74	31	1797.24	2011-09-25 13:13:00	0.627834	4	0.588038	3	0.226516	2	432	2.58
3	12349	18	73	1757.55	2011-11-21 09:51:00	0.281930	2	0.341277	2	0.232994	2	222	2.00
4	12350	309	17	334.40	2011-02-02 16:01:00	0.949560	5	0.759949	4	0.711708	4	544	4.15
5	12352	35	95	1545.41	2011-11-03 14:37:00	0.432901	3	0.271402	2	0.261453	2	322	2.15

Figure 13: Dataset after RFM Analysis

5.5 RFM Class

A new class column is created with the following classes: good - 'g', bad - 'b', medium - 'm' based on their weighted RFM score under a certain threshold value for each class.

```
# Function used in the creation of a class column out of RFM score
# Good, medium, and bad customers based on the weighted RFM score
def myClass(num):
    if num >=1 and num <2:
        return 'g'              ## good customers
    elif num >=2 and num <4:
        return 'm'              ## medium customers
    else:
        return 'b'              ## bad customers

################################################################################
# create the class column (RFM as a classification problem here, it could also
# be posed as a regression problem)
rfm['class'] = list(map(myClass, rfm['rfm_score']))
```

Figure 14: Cluster creation - g,b,m

5.6 Cluster Classification & Forecasting

A time-based evaluation of RF & GBM classifier on the RFM dataset to determine the best month for forecasting customer classes is conducted in this code section. The code

iterates 10 times (from 1 to 10) to evaluate the classifier's performance for different time-periods, where a two-month training period and one-month test or forecasting period is considered. 'clf.fit(X_train, y_train)' is used to train the model on the train dataset. The classification report generates the evaluation metrics such as, precision, recall and F1-scores. A table or plot can be used to visualize the F1-scores and accuracies for each month. Based on the results, the month with the highest F1-score or accuracy is identified as the best month for forecasting.

```
trainData = rfm[ (rfm['lastPurchase'] >= baseDate) & (rfm['lastPurchase'] <= baseDate+relativedelta(months=2)) ]    ## take 2m data
testData = rfm[ (rfm['lastPurchase'] > baseDate+relativedelta(months=2)) & (rfm['lastPurchase'] < baseDate+relativedelta(months=3))
#
print('Train data Min and max dates are:', trainData['lastPurchase'].min(), 'to', trainData['lastPurchase'].max())
print('Test data Min and max dates are:', testData['lastPurchase'].min(), 'to', testData['lastPurchase'].max())
#
X_train = trainData[['r_score', 'f_score', 'm_score']]
y_train = trainData['class']
X_test = testData[['r_score', 'f_score', 'm_score']]
y_test = testData['class']
#
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
#
report = classification_report(y_test, y_pred, output_dict=True)
print(classification_report(y_test, y_pred))
#
f1_score_rf = report['weighted avg']['f1-score']
f1_scores_rf.append(f1_score_rf)
```

Figure 15: Cluster Forecasting

```
Train data Min and max dates are: 2011-07-01 10:47:00 to 2011-08-31 16:28:00
Test data Min and max dates are: 2011-09-01 09:57:00 to 2011-09-30 15:52:00
              precision    recall  f1-score   support

           b       0.97      0.97      0.97       119
           g       1.00      0.97      0.98        33
           m       0.98      0.99      0.98       242

    accuracy                           0.98       394
   macro avg       0.98      0.97      0.98       394
weighted avg       0.98      0.98      0.98       394
```

Figure 16: Classification Report for September forecast